HyperLogLog: Analysis and implementation of an improved algorithm

Dequeker Chloé, Ziat Ghiles

February 2015

Contents

1 Introduction

2 cardinality estimation problem

Finding the number of distinct elements in a data set with duplicates is a well-known problem which applies in many fields.

The naive solution to this problem is to examine for each element of the data stream its belonging to a data structure $\mathcal{D}$. If $\mathcal{D}$ does not contain the element we add it to the data structure. At the end of the process, the cardinality of the data stream is equal to the size of $\mathcal{D}$.

This solution gives the exact answer but it is easy to see that it scales very badly as the size of the data stream grows.

In order to resolve this problem, several alogorithms have been proposed. These include LinearCounting and HyperLogLog which are the two bases of the studied alogorithm.

3 LinearCounting and HyperLogLog

3.1 LinearCounting

3.2 HyperLogLog

4 HyperLogLog++

4.1 transition to 64 bits

4.2 Bias estimation and correction

For a given configuration of the algorithm, the observed bias is dependant on the cardinality estimated. From this observation, we implement a correction method.

4.3 Memory optimization

4.3.1 Sparse representation

4.3.2 Dense representation

4.3.3 Varint encoding

Since the temporary set used in the sparse representation is merged with the list before it gets too large, performing a compression on it is not as interesting then on the sorted list. We'll try to reduce the memory usage of it by playing on two points:

- Using fixed-size integers as it is common practice in many langages may here result in a waste of memory space.

- Since the manipulated list is sorted, we can take advantage of this information.

5   Conclusion